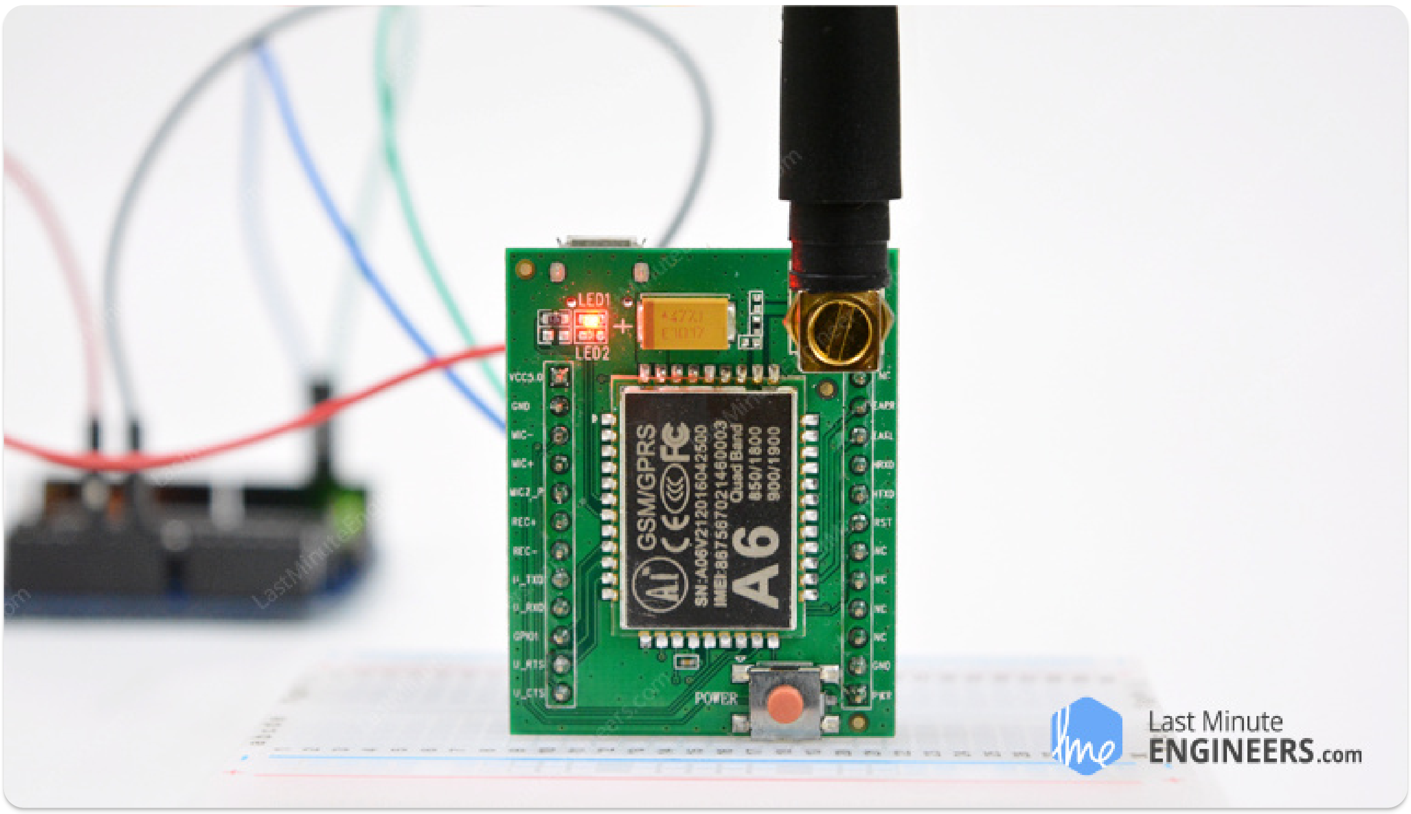
Last Minute ENGINEERS

Send Receive SMS & Call with A6 GSM Module & Arduino

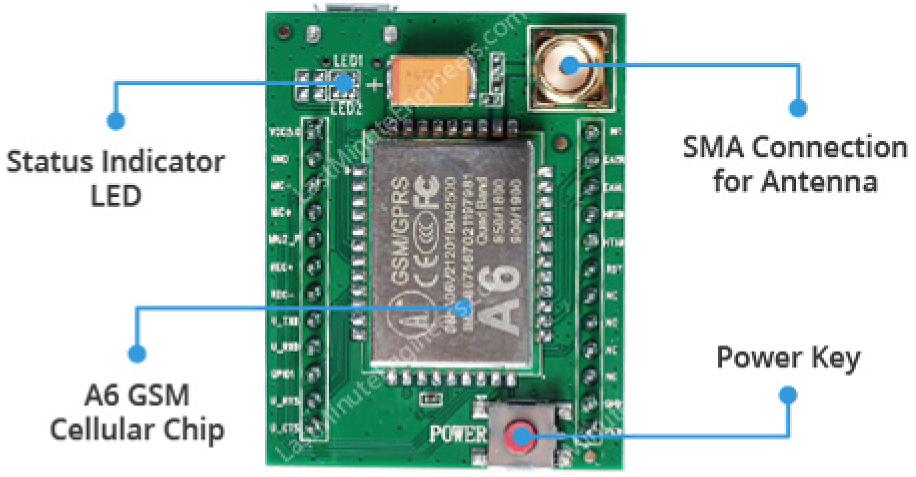


Whether you want to listen to what happens in your house that's miles away from you or activate sprinkler system in your garden just with a silent call; Then A6 GSM/GPRS module serves as a solid launching point for you to get you started with IoT!

A6 GSM/GPRS module is a miniature GSM modem, which can be integrated into a great number of IoT projects. You can use this module to accomplish almost anything a normal cell phone can; SMS text messages, Make or receive phone calls, connecting to internet through GPRS, TCP/IP, and more! To top it off, the module supports quad-band GSM/GPRS network, meaning it works pretty much anywhere in the world.

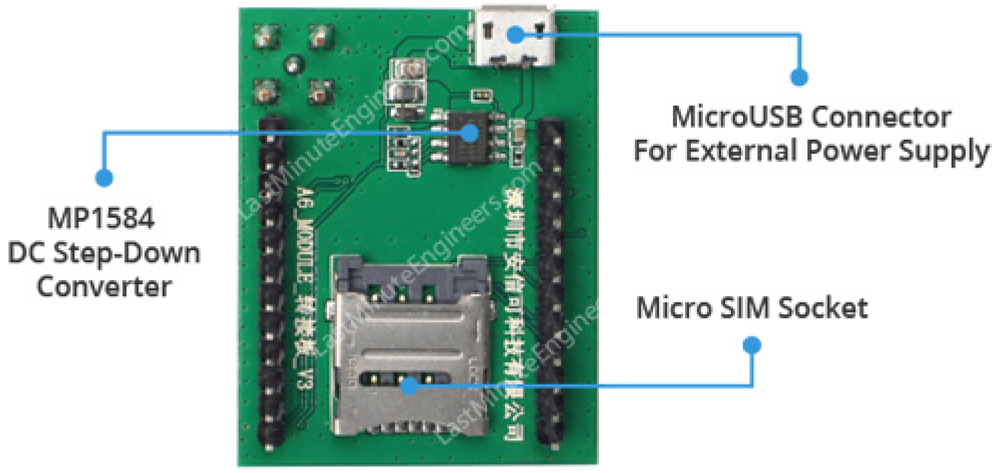
Hardware Overview of A6 GSM/GPRS module

necessary data pins of A6 GSM chip are broken out to a 0.1" pitch headers.



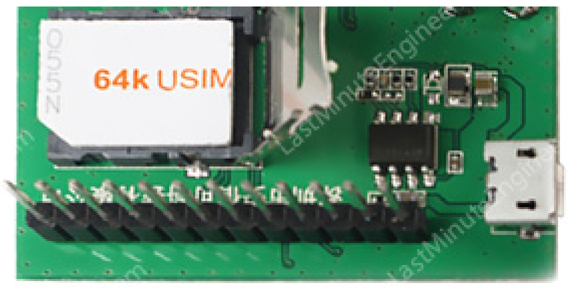
The module needs an external antenna for any kind of voice or data communications as well as some SIM commands. So, it usually comes with a small **duck antenna having 2 dBi gain** and 50Ω impedance which provides great coverage even if your project is indoors.

There is a **Power button** provided to turn the module ON/OFF manually, though you can do this programmatically. The status of the module is indicated by an LED on the top right side of the Module.


MicroUSB Connector For External Power Supply
MP1584 DC Step-Down Converter
Micro SIM Socket

Though the module can work on **5V**, the operating voltage of the chip is from **3.3V to 4.2V**. To keep supply voltage safe at 4.1V, the module comes with a high frequency step-down switching regulator MP1584 from Monolithic Power Systems – capable of handling load currents up to 3A.

The module can also be powered through a **micro USB connector**. You can just grab your cell phone’s wall charger (rated 5V 2A) and power up the module. How cool is that!



There's a SIM socket on the back! Any activated, **2G micro SIM card** would work perfectly. The workings of the SIM card socket can take some getting used to. To unlock the latch, push the top part of the assembly towards micro USB connector, and then lift it up. Place the SIM card into the bottom part of the socket with the SIM's notch pointing away from the micro USB connector. Then fold the arm back into the body of the socket, and gently push it forward towards the "LOCK" position.

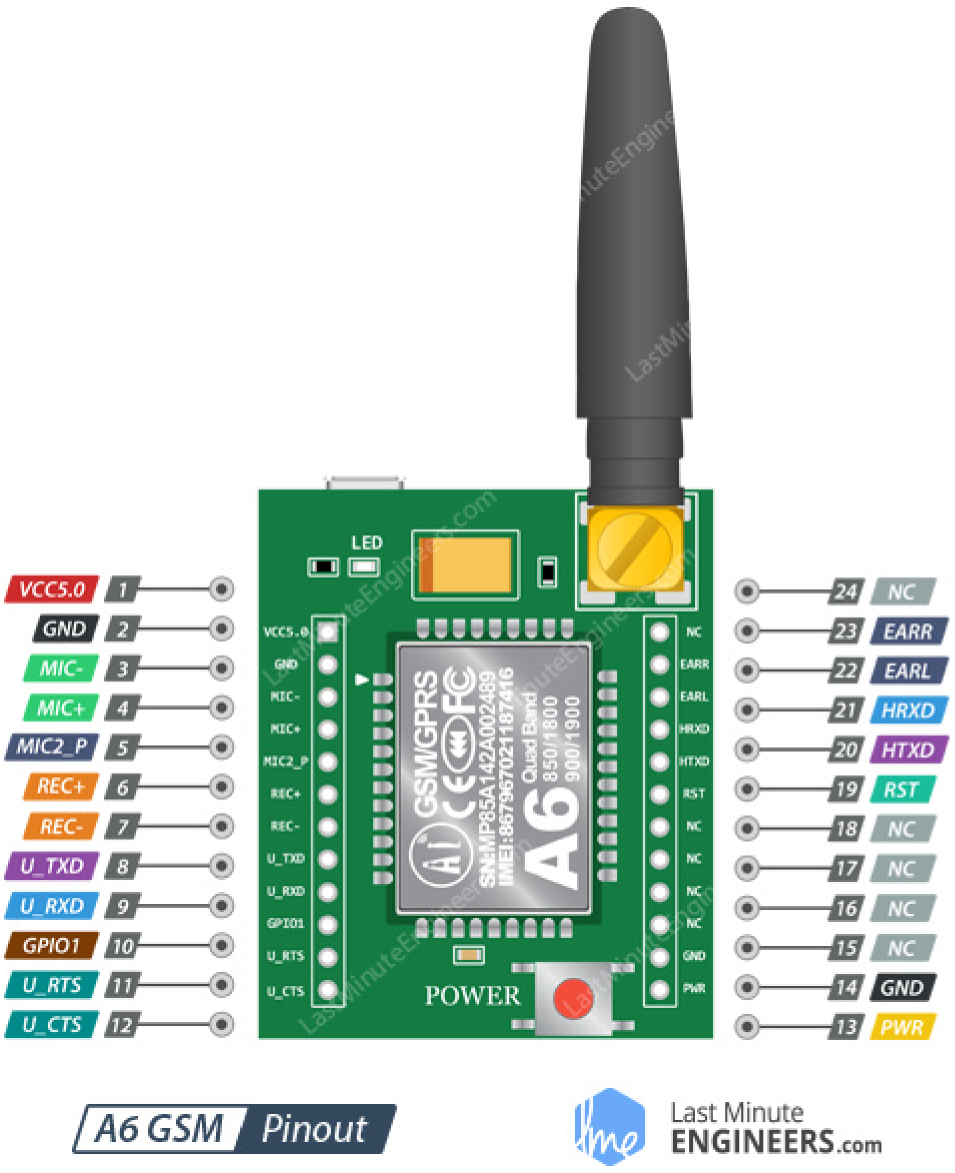
A6 GSM cellular chip measures less than the size of a postage stamp but packs a surprising amount of features into its little frame. Some of them are listed below:

- Supports Quad-band: GSM850, EGSM900, DCS1800 and PCS1900
- Connect onto any global GSM network with any 2G SIM
- Make and receive voice calls using an external 8Ω speaker & electret microphone
- Facility to connect 4-pole TRRS mic and headset
- Send and receive Voice calls and SMS messages
- Class 10 GPRS with 85.6Kbps download speed and 42.8Kbps upload speed
- Consumes less than 3mA in standby mode
- 8V UART port level – compatible with Arduino, Raspberry-Pi
- Transmit Power:
    - Class 4 (2W) for GSM850/EGSM900
    - Class 1 (1W) for DCS1800/PCS1900
- Supports serial-based AT Command Set
- Accepts Micro SIM Card

For more information about A6 GSM Cellular chip, check out this datasheet.

A6 GSM Module Pinout

The A6 GSM module has total 24 pins that interface it to the outside world. The connections are as follows:



VCC supplies power for the module. Connect this to any external power source rated 5V 2A.

GND is the Ground Pin and needs to be connected to GND pin on the Arduino.

MIC± is a differential microphone input. The two microphone pins can be connected directly to these pins.

MIC2_P pin is used to interface 4-pole TRRS MIC

U_TxD (Transmitter) pin is used for serial communication.

U_RxD (Receiver) pin is used for serial communication.

GPIO1 is used to control the module to enter low-power mode.

U_RTS (Request to Send) is UART flow control pin allow the receiver and the transmitter to alert each other to their state.

U_CTS (Clear to Send) is UART flow control pin allow the receiver and the transmitter to alert each other to their state.

EAR_R is used to interface 4-pole TRRS Headset

EAR_L is used to interface 4-pole TRRS Headset

HST_RXD HOST UART is a debug UART, which is used for downloading, calibrating, trace and so on. It doesn't support any AT command. This interface is only used when debugging

HST_TXD UART is a debug UART, which is used for downloading, calibrating, trace and so on. It doesn't support any AT command. This interface is only used when debugging

RST (Reset) is a hard reset pin. If you absolutely got the module in a bad space, pull this pin low for 100ms to perform a hard reset.

NC Not Connected

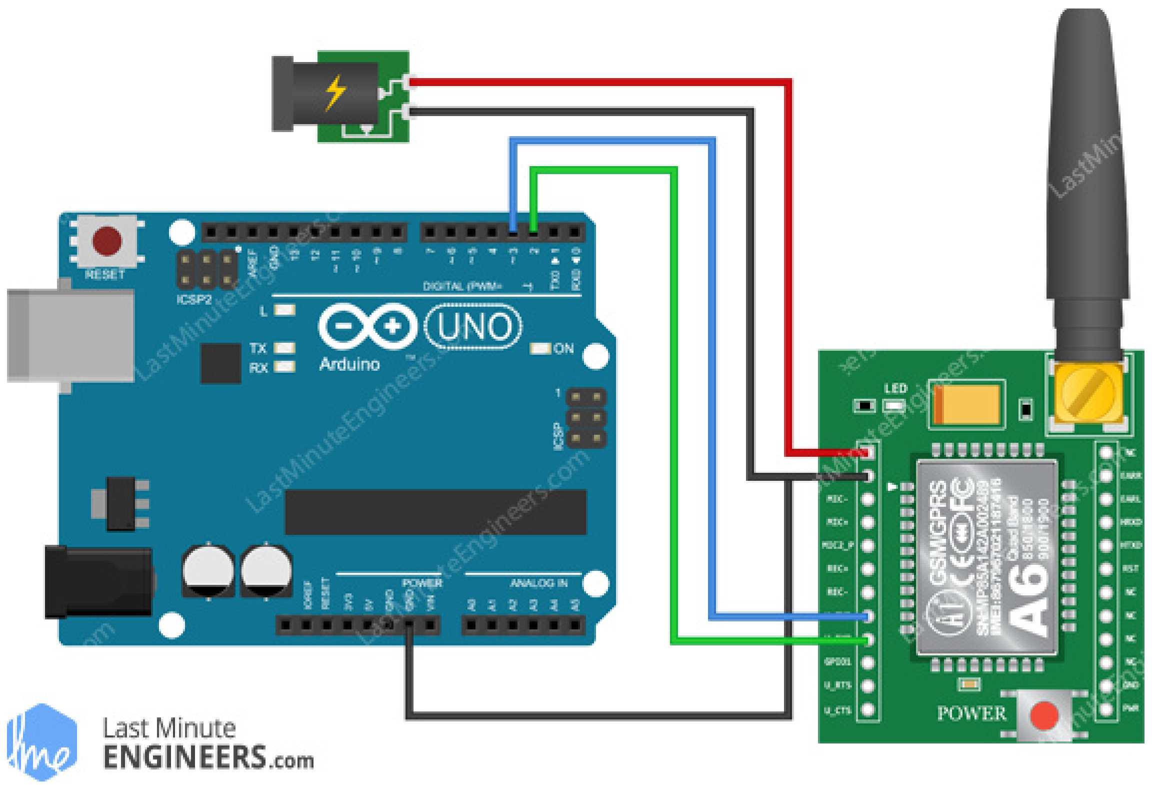
PWR pin is used for turning module ON/OFF programmatically. For doing this you must pull it HIGH for a moment (less than 500 ms or around).

Wiring – Connecting A6 GSM module to Arduino UNO

Now that we know everything about the module, we can begin hooking it up to our Arduino!

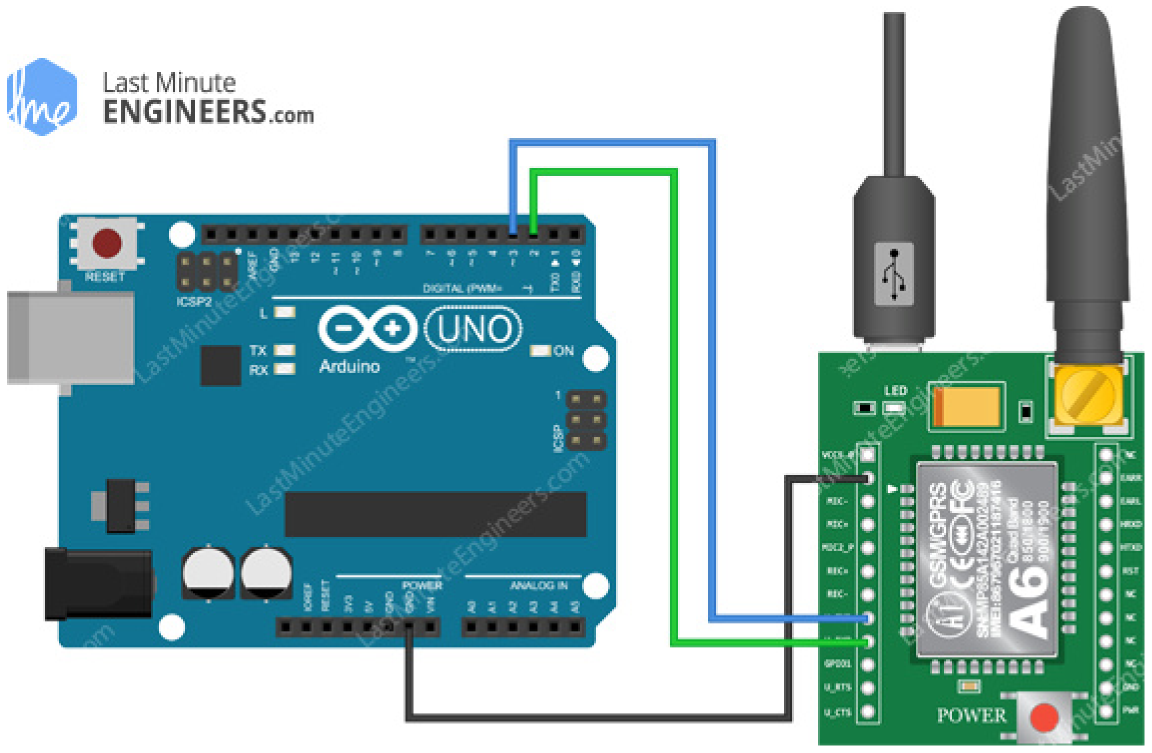
To start with, connect U_TxD and U_RxD pin on module to digital pin#3 and #2 on Arduino as we'll be using software serial to talk to the module.

Connect VCC pin on module to external power supply rated 5V 2A. Do not be tempted to connect this pin to 5V supply on Arduino, as the module will not work due to the lack of supply current. Now connect all the ground in the circuit.



Wiring A6 GSM GPRS Module With Arduino UNO Using External Power Supply

You can also power up the module through 5V 2A wall adapter but make sure that GND pin on the Arduino is connected to GND pin on module.



Wiring A6 GSM GPRS Module With Arduino UNO Using USB Power Supply

WARNING

You should be very careful to not to disconnect the GND before the VCC and always connect GND before VCC. Otherwise the module can use the low voltage serial pins as ground and can get destroyed instantly.

Arduino Code – Testing AT Commands

For sending AT commands and communicating with the A6 module, we will use the serial monitor. The sketch below will enable the Arduino to communicate with the A6 module on serial monitor. Before we proceed with detailed breakdown of code, connect your Arduino to PC, compile below code and upload it to the Arduino.

Once you open a serial monitor, **make sure that 'Both NL & CR' option is selected!**

```
  updateSerial();
  mySerial.println("AT+CCID"); //Read SIM information to confirm
whether the SIM is plugged
  updateSerial();
  mySerial.println("AT+CREG?"); //Check whether it has registered in
the network
  updateSerial();
}

void loop()
{
  updateSerial();
}

void updateSerial()
{
  delay(500);
  while (Serial.available())
  {
    mySerial.write(Serial.read());//Forward what Serial received to
Software Serial Port
  }
  while(mySerial.available())
  {
```

The sketch starts by including a **SoftwareSerial.h** library and initializing it with the Arduino pins to which Tx and Rx of A6 module is connected.

```
#include <SoftwareSerial.h>

//Create software serial object to communicate with A6
SoftwareSerial mySerial(3, 2); //A6 Tx & Rx is connected to Arduino #3
& #2
```

In setup function: we initialize a serial communication link between Arduino, Arduino IDE and A6 module at a baud rate of 9600.

```
//Begin serial communication with Arduino and Arduino IDE (Serial
Monitor)
  Serial.begin(9600);

  //Begin serial communication with Arduino and A6
  mySerial.begin(9600);
```

Now that we have established a basic connection, we will try to communicate with the A6 module by sending AT commands.

AT – It is the most basic AT command. It also initializes Auto-baud'er. If it works you should see the AT characters echo and then OK, telling you it's OK and it's understanding you correctly! You can then send some commands to query the module and get information about it such as

AT+CSQ – Check the 'signal strength' – the first # is dB strength, it should be higher than around 5. Higher is better. Of course it depends on your antenna and location!

AT+CCID – get the SIM card number – this tests that the SIM card is found OK and you can verify the number is written on the card.

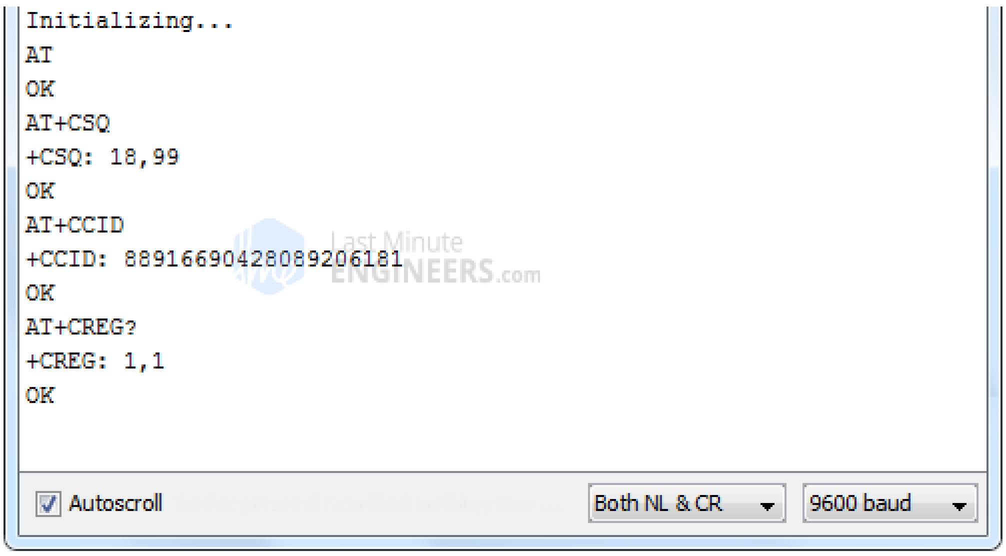
AT+CREG? Check that you're registered on the network. The second # should be 1 or 5. 1 indicates you are registered to home network and 5 indicates roaming network. Other

```
  mySerial.println("AT"); //Once the handshake test is successful, it 
will back to OK
  updateSerial();
  mySerial.println("AT+CSQ"); //Signal quality test, value range is 0-
31 , 31 is the best
  updateSerial();
  mySerial.println("AT+CCID"); //Read SIM information to confirm 
whether the SIM is plugged
  updateSerial();
  mySerial.println("AT+CREG?"); //Check whether it has registered in 
the network
  updateSerial();
```

In the looping part of the code, we call custom function called **updateSerial()** which continuously waits for any inputs from the serial monitor and send it to the A6 module through the D2 pin (Rx of module). It also continuously reads the D3 pin (Tx of module) if the A6 module has any responses.

```
void updateSerial()
{
  delay(500);
  while (Serial.available())
  {
    mySerial.write(Serial.read());//Forward what Serial received to 
Software Serial Port
  }
  while(mySerial.available())
  {
    Serial.write(mySerial.read());//Forward what Software Serial 
received to Serial Port
  }
}
```

You should see below output on serial monitor.



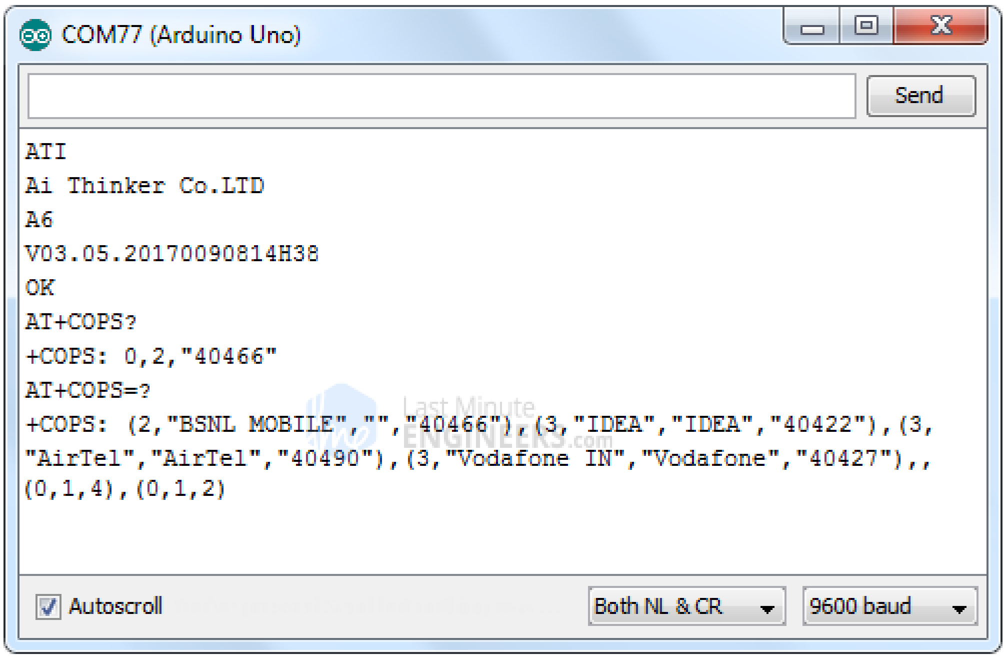
```
Initializing...
AT
OK
AT+CSQ
+CSQ: 18,99
OK
AT+CCID
+CCID: 89916690428089206181
OK
AT+CREG?
+CREG: 1,1
OK
```

You are now free to send any commands through serial monitor like below which gives more information about network connection:

ATI – Get the module name and revision

AT+COPS? – Check which network you are connected to, in this case 40466 (BSNL)

AT+COPS=? – Return the list of operators present in the network.



```
ATI
Ai Thinker Co.LTD
A6
V03.05.20170090814H38
OK
AT+COPS?
+COPS: 0,2,"40466"
AT+COPS=?
+COPS: (2,"BSNL MOBILE","","40466"),(3,"IDEA","IDEA","40422"),(3,
"AirTel","AirTel","40490"),(3,"Vodafone IN","Vodafone","40427"),,
(0,1,4),(0,1,2)
```

For more A6 AT Commands, please refer this document.

A6 GSM AT Commands

phone number you wish. Before trying the sketch out, you need to enter the phone number. Search for the string ZZxxxxxxxxxx and replace ZZ with county code and xxxxxxxxxx with the 10 digit phone number.

```
#include <SoftwareSerial.h>

//Create software serial object to communicate with A6
SoftwareSerial mySerial(3, 2); //A6 Tx & Rx is connected to Arduino
#3 & #2

void setup()
{
  //Begin serial communication with Arduino and Arduino IDE (Serial
Monitor)
  Serial.begin(9600);

  //Begin serial communication with Arduino and A6
  mySerial.begin(9600);

  Serial.println("Initializing...");
  delay(1000);

  mySerial.println("AT"); //Once the handshake test is successful, it
will back to OK
  updateSerial();

  mySerial.println("AT+CMGF=1"); // Configuring TEXT mode
  updateSerial();
  mySerial.println("AT+CMGS=\"+ZZxxxxxxxxxx\"");//change ZZ with
country code and xxxxxxxxxxx with phone number to sms
```

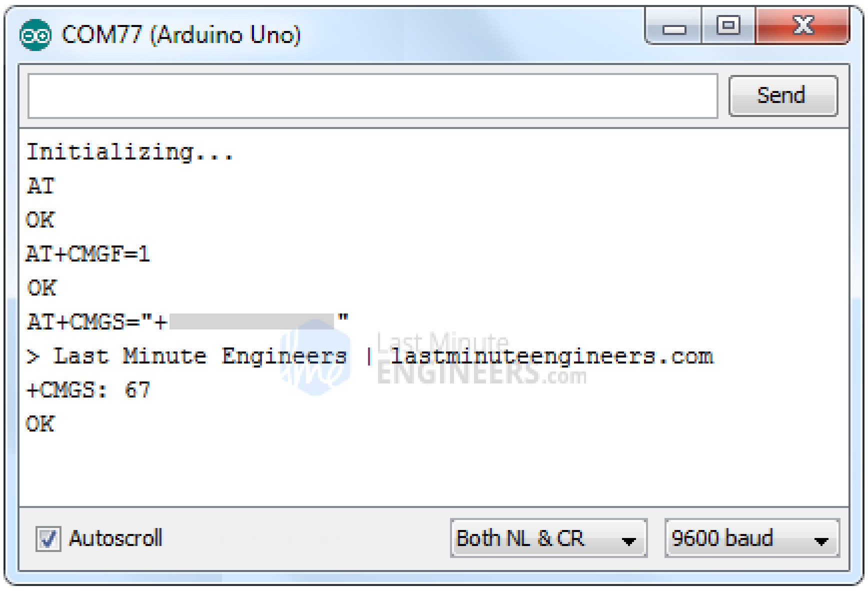
The sketch is almost same as earlier except below code snippet. Once the connection is established, we send below AT commands:

AT+CMGF=1 – Selects SMS message format as text. Default format is **P**rotocol **D**ata **U**nit (PDU)

AT+CMGS=+ZZxxxxxxxxxx – Sends SMS to the phone number specified. The text message entered followed by a 'Ctrl+z' character is treated as SMS. 'Ctrl+z' is actually a

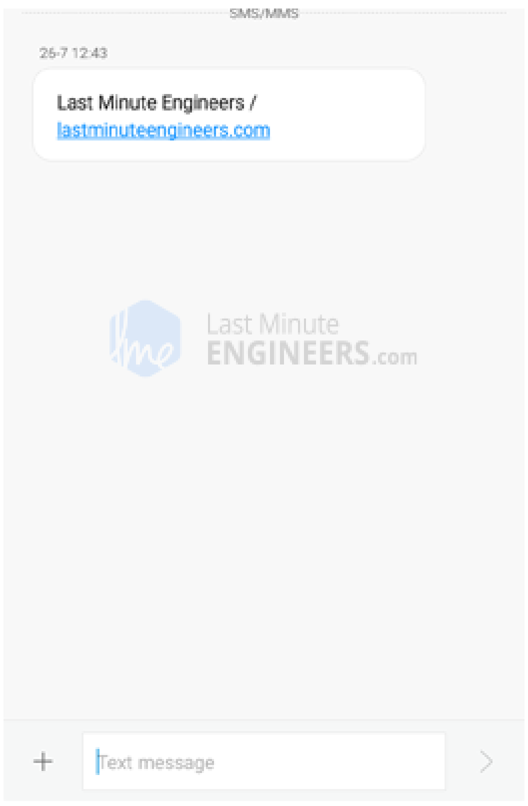
```
  mySerial.println("AT+CMGF=1"); // Configuring TEXT mode
  updateSerial();
  mySerial.println("AT+CMGS=\"+ZZxxxxxxxxxx\"");//change ZZ with
country code and xxxxxxxxxxx with phone number to sms
  updateSerial();
  mySerial.print("Last Minute Engineers | lastminuteengineers.com");
//text content
  updateSerial();
  mySerial.write(26);
```

The loop is kept empty as we want to send SMS only once. If you wish to send SMS one more time, just hit the RESET key on your Arduino. Below screenshot shows SMS sent from A6 GSM module.



```
Initializing...
AT
OK
AT+CMGF=1
OK
AT+CMGS="+[redacted]"
> Last Minute Engineers | lastminuteengineers.com
+CMGS: 67
OK
```

Last Minute ENGINEERS


SMS/MMS

25-7 12:43

Last Minute Engineers / lastminuteengineers.com

Arduino Code – Reading SMS

Now let’s program our Arduino to read incoming messages. This sketch is very useful when you need to trigger an action when a specific SMS is received. For example, when the Arduino receives an SMS, you can instruct it to turn on or off a relay. You got the idea!

```
#include <SoftwareSerial.h>

//Create software serial object to communicate with A6
SoftwareSerial mySerial(3, 2); //A6 Tx & Rx is connected to Arduino
#3 & #2

void setup()
{
  //Begin serial communication with Arduino and Arduino IDE (Serial
Monitor)
  Serial.begin(9600);

  //Begin serial communication with Arduino and A6
  mySerial.begin(9600);
  mySerial.println("AT"); //Once the handshake test is successful, it
will back to OK
  updateSerial();

  mySerial.println("AT+CMGF=1"); // Configuring TEXT mode
  updateSerial();
  mySerial.println("AT+CNMI=1,2,0,0,0"); // Decides how newly arrived
```

The sketch is similar as earlier except below code snippet. Once the connection is established, we send below AT commands:

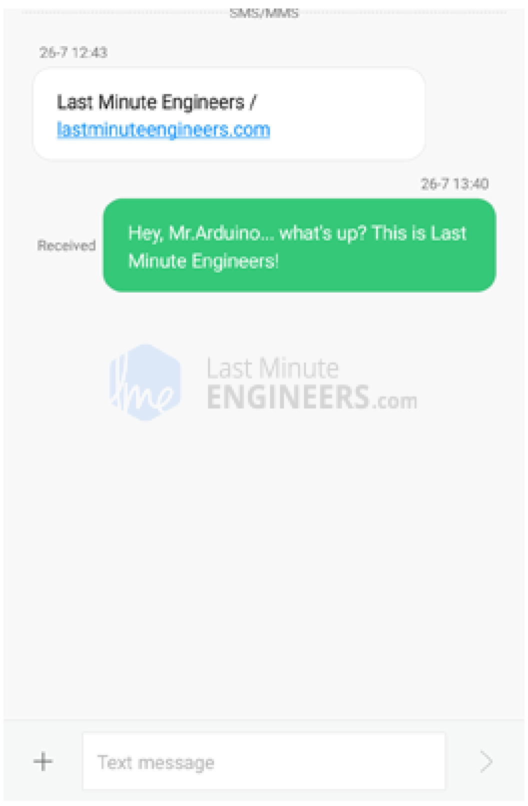
AT+CMGF=1 – Selects SMS message format as text. Default format is **P**rotocol **D**ata **U**nit (PDU)

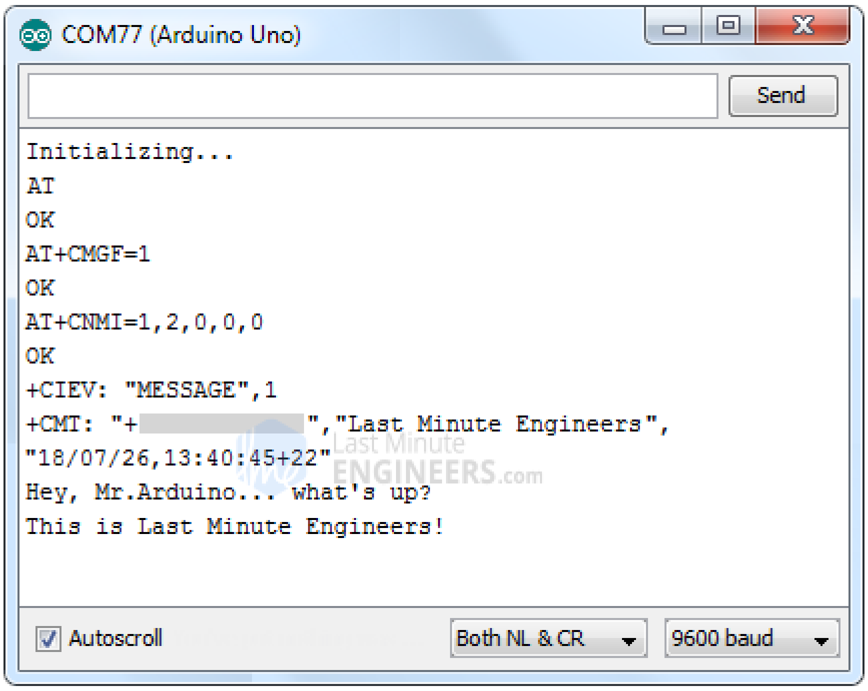
AT+CNMI=1,2,0,0,0 – specifies how newly arrived SMS messages should be handled. This way you can tell the A6 module either to forward newly arrived SMS messages directly to the PC, or to save them in message storage and then notify the PC about their locations in message storage.

Its response starts with **+CMT:** All the fields in the response are comma-separated with first field being phone number. The second field is the name of person sending SMS. Third field is a timestamp while forth field is the actual message.

```
  mySerial.println("AT+CMGF=1"); // Configuring TEXT mode
  updateSerial();
  mySerial.println("AT+CNMI=1,2,0,0,0"); // Decides how newly arrived
SMS messages should be handled
  updateSerial();
```

Note that **this time we have NOT kept the loop function empty** as we are polling for newly arrived SMS messages. Once you send the SMS to A6 GSM module, you will see below output on serial monitor.


SMS/MMS
26-7 12:43
Last Minute Engineers / lastminuteengineers.com
26-7 13:40
Received
Hey, Mr.Arduino... what's up? This is Last Minute Engineers!
Text message



```
Initializing...
AT
OK
AT+CMGF=1
OK
AT+CNMI=1,2,0,0,0
OK
+CIEV: "MESSAGE",1
+CMT: "+[redacted]","Last Minute Engineers",
"18/07/26,13:40:45+22"
Hey, Mr.Arduino... what's up?
This is Last Minute Engineers!
```

Expanding Arduino SoftwareSerial Buffer Size

If your message is long enough just like ours, then you'll probably receive it with some missing characters. This is not because of a faulty code. Your SoftwareSerial receive buffer is getting filled up and discarding characters. You are not reading fast enough from the buffer.

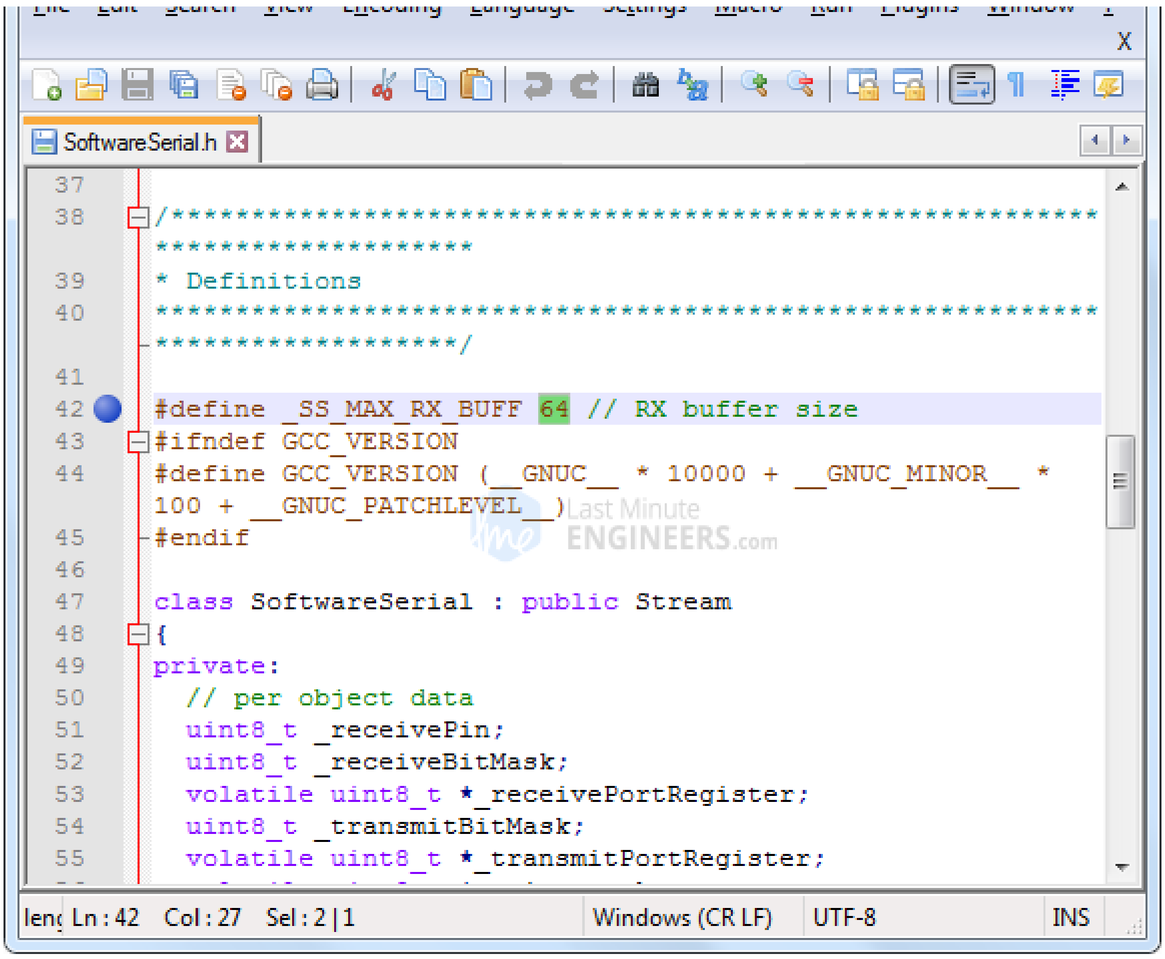
On a Windows PC, go to C:\Program Files (x86) -> Arduino -> hardware -> Arduino -> avr -> libraries -> SoftwareSerial (-> src for newer version of Arduino IDE) Open SoftwareSerial.h and change the line:

```
// RX buffer size
#define _SS_MAX_RX_BUFF 64
```

to

```
// RX buffer size
#define _SS_MAX_RX_BUFF 512
```

Save the file and try your sketch again.



```
/**************************************************************************************
* Definitions
**************************************************************************************/

#define _SS_MAX_RX_BUFF 64 // RX buffer size
#ifndef GCC_VERSION
#define GCC_VERSION (__GNUC__ * 10000 + __GNUC_MINOR__ * 100 + __GNUC_PATCHLEVEL__)
#endif

class SoftwareSerial : public Stream
{
private:
  // per object data
  uint8_t _receivePin;
  uint8_t _receiveBitMask;
  volatile uint8_t *_receivePortRegister;
  uint8_t _transmitBitMask;
  volatile uint8_t *_transmitPortRegister;
```

Expanding Arduino SoftwareSerial Buffer Size

Arduino Code – Making Call

Now let's program our Arduino to make call. This sketch is very useful when you want your Arduino to make an SOS/distress call in case of emergency like temperature being exceeded or someone breaks into your house. You got the idea!

Before trying the sketch out, you need to enter the phone number. Search for the string ZZxxxxxxxxxx and replace ZZ with county code and xxxxxxxxxx with the 10 digit phone number.

```
#include <SoftwareSerial.h>

//Create software serial object to communicate with A6
SoftwareSerial mySerial(3, 2); //A6 Tx & Rx is connected to Arduino
#3 & #2
```

```
Monitor)
  Serial.begin(9600);

  //Begin serial communication with Arduino and A6
  mySerial.begin(9600);

  Serial.println("Initializing...");
  delay(1000);

  mySerial.println("AT"); //Once the handshake test is successful, i
t will back to OK
  updateSerial();

  mySerial.println("ATD+ZZxxxxxxxxxx"); //  change ZZ with country
code and xxxxxxxxxxx with phone number to dial
  updateSerial();
  delay(20000); // wait for 20 seconds
```

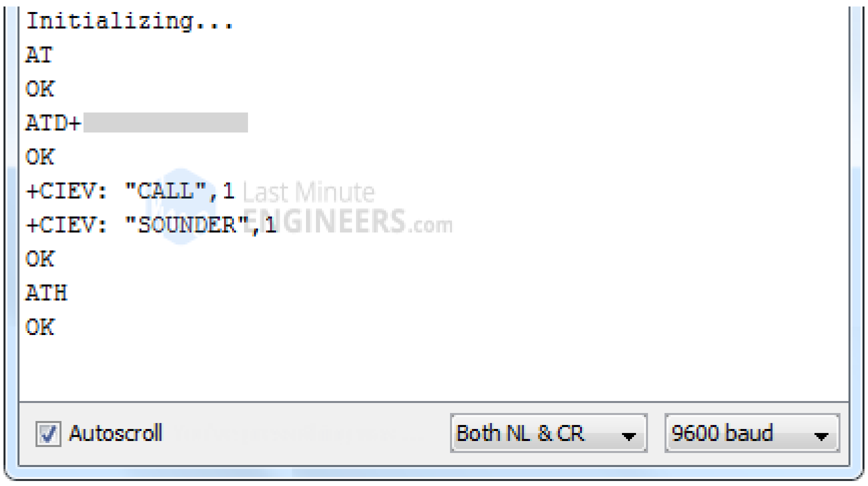
To place a call following AT commands are used:

ATD+ZZxxxxxxxxxx; – Dials a specified number.

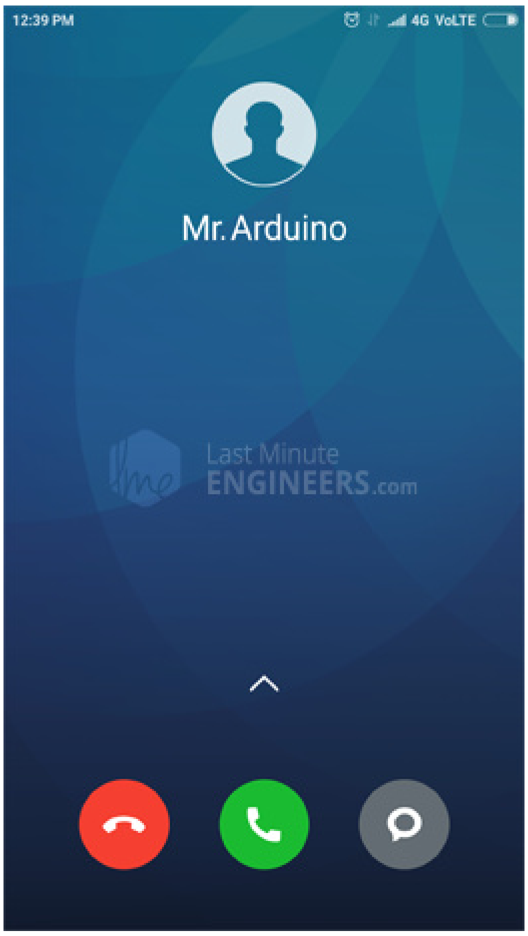
ATH – Hangs up the call

```
  mySerial.println("ATD+ZZxxxxxxxxxx"); //  change ZZ with country
code and xxxxxxxxxxx with phone number to dial
  updateSerial();
  delay(20000); // wait for 20 seconds...
  mySerial.println("ATH"); //hang up
  updateSerial();
```

Below screenshot shows call made from A6 GSM module.



```
Initializing...
AT
OK
ATD+[redacted]
OK
+CIEV: "CALL",1
+CIEV: "SOUNDER",1
OK
ATH
OK
```



Arduino Code – Receiving Call

Receiving call doesn't require any special code; you just have to keep listening to the A6 module. Yet, you may find this sketch very useful, when you need to trigger an action when a call from specific phone number is received.

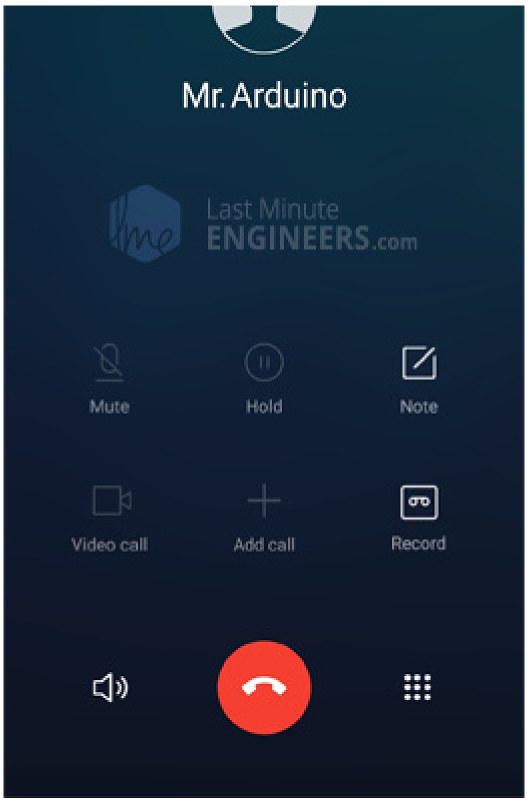
```
//Create software serial object to communicate with A6
SoftwareSerial mySerial(3, 2); //A6 Tx & Rx is connected to Arduino
#3 & #2

void setup()
{
  //Begin serial communication with Arduino and Arduino IDE (Serial
Monitor)
  Serial.begin(9600);

  //Begin serial communication with Arduino and A6
  mySerial.begin(9600);

  Serial.println("Initializing...");
}

void loop()
{
  updateSerial();
}

void updateSerial()
{
  delay(500);
```

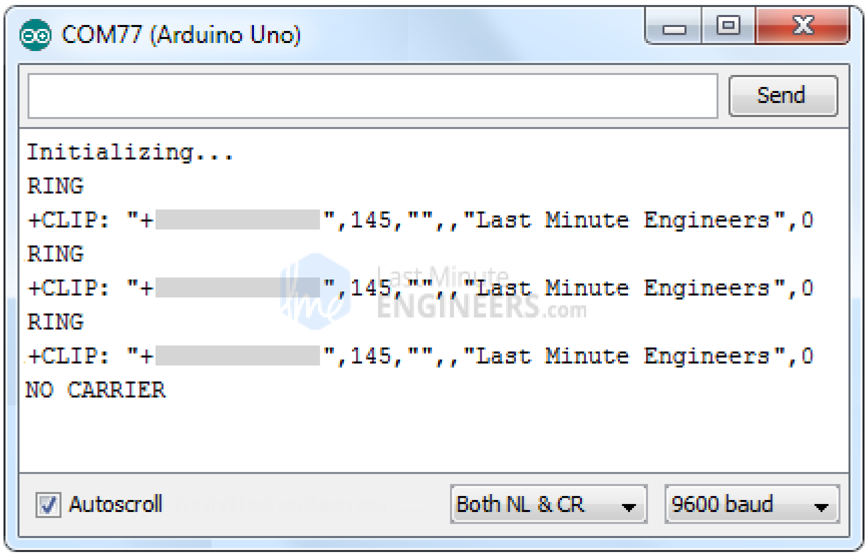
Incoming call is usually indicated by 'RING' on serial monitor followed by phone number and caller ID. To accept/hang a call following AT commands are used:

ATA – Accepts incoming call.

ATH – Hangs up the call. On hanging up the call it sends NO CARRIER on the serial monitor indicating call couldn't connect.

Below output on serial monitor shows call received by A6 GSM module.




COM77 (Arduino Uno)

Send

```
Initializing...
RING
+CLIP: "+[redacted]",145,"",,"Last Minute Engineers",0
RING
+CLIP: "+[redacted]",145,"",,"Last Minute Engineers",0
RING
+CLIP: "+[redacted]",145,"",,"Last Minute Engineers",0
NO CARRIER
```

[✓] Autoscroll   Both NL & CR   9600 baud

€0.94

€48

MCP1703-3302E/DB
MICROCHIP TECHNOLOGY,...

€0.78

ATMEGA328P-AU
MICROCHIP (ATM

€2.18

Share

Disclaimer Privacy Policy